ISSN 0956-9979 MAY 1996

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Defending the macros.** One of the most publicised virus 'events' of last year was the appearance of macro viruses, which have now spread world-wide. Close behind it ran the vendors, with various fixes and cures. Who has what, and how do they perform? See p.10 for our evaluation.

• **How much does 'crying wolf' cost?** If there were a virus incident in your company, how much would it be likely to cost? And if it were to be a false alarm, have you implemented adequate policies to be able to pinpoint it immediately? Turn to p.16 for one company's experiences.

• **Making outlaws popular?** *VB* has just learned that the infamous Mark Ludwig has released an 'update' to his CD-ROM virus collection, first published some eighteen months ago. See News page (p.3) for more information.

CONTENTS

EDITORIAL

Guarding Against Folly

Readers of *VB* will have noticed, over the last few years, the not-infrequent references to the methods and ethics of the distribution of virus code. Distribution of such code is something against which anti-virus people campaign, often falling foul of free-speech issues along the way. At the *National Computer Security Association's* April conference, *IVPC'96*, the issues and their difficulties were demonstrated in an uncomfortably close-to-home and pertinent fashion.

"the apparent lack of concern with which the issue was viewed ... is more than slightly worrying"

At this, the *International Virus Prevention Conference*, the *NCSA* offered delegates a 'book table', which featured the latest books on the virus problem and general computer security, in addition to old favourites – an excellent idea. The books were taken from those listed in the *NCSA* catalogue, and offered for sale at a discount on the normal prices.

Amongst the titles was an unremarkable-looking work called *Virus Detection and Elimination*, written by a Dane, Rune Skardhamar. The title blended perfectly with all the other virus-related books on the stall; alas, its contents did not.

I was advised, late in the conference, to take a look at the book, and went to the stall to browse. My initial impressions were poor – it is badly written, and contains numerous factual errors. Such statements as 'Remember, no infection can occur by simple scanning for viruses, provided the scanner is not itself infected' are inaccurate; indeed, downright dangerous. The crunch, however, is that amongst the usual low-grade virus and anti-virus information, the book offers virus code. Complete viruses – and this is where ethics become an issue...

Whilst it is true that at least some of the virus code presented in Skardhamar's book does not work, it is equally true that it can, with a minimum of effort, be made to produce a functioning virus. However, this is not the most significant issue. The point is this: if an organisation such as the *NCSA*, heavily involved as it purports to be in promoting a sensible attitude to distribution both of virus code and virus-writing manuals, can miss a book as obvious in its content as this, what hope is there for organisations with a less specific remit? How is a book-shop supposed to know that it would be a bad idea to sell such a title if the fact escaped even the *NCSA's* notice?

Or did it? The *NCSA* was told of the dubious nature of the book in question – to my knowledge, twice during *IVPC'96*. Some *NCSA* staff members were horrified (Mich Kabay, the *NCSA's* Director of Education, foremost amongst them), and the book was at one point removed from the stall, only to be reinstated later in the conference. I am reliably informed that one *NCSA* staff member even used the stale argument: 'If we don't sell it, someone else will'. George Smith (author of the *American Eagle Publications* book *The Virus Creation Labs*, and producer of the *Crypt* newsletter) was one of the people to point out the book's contents during the course of the conference, yet it was still on sale at the very end of the conference, when I obtained my copy.

Even this, however, was not all: the book had previously been reviewed by Smith, who described its contents in such a way as to leave no doubt that the book would be unsuitable for sale by the *NCSA*, in an issue of the *Crypt* newsletter which was available well before the conference from the *NCSA's* own *CompuServe* forum.

By coincidence, whilst I was at the stall looking at the book, at the end of the conference, *NCSA* President Dr Peter Tippett walked past. I took the opportunity to ask him whether he was aware that the book contained virus source code, to which he replied: 'What are they going to do, scan it in?'

The *NCSA* is to be commended on having now removed the book from its catalogue and withdrawn it from sale; however, the apparent lack of concern with which the issue was viewed at the conference, and the initial reaction of the *NCSA's* figurehead and spokesman, is more than slightly worrying.

Juvenal's question, 'Quis custodiet ipsos custodes' (Who is to guard the guards themselves?) has never, alas, been more apt.

NEWS

Wanted: A Fistful of Dollars

Cheyenne Software Inc has announced that on 15 April 1996 the company's board of directors unanimously rejected what amounts to a hostile takeover bid from *McAfee Associates*.

ReiJane Huai, *Cheyenne's* President and CEO, had this to say: '*Cheyenne's* Board of Directors and management are keenly focused on increasing shareholder value, and we have carefully considered *McAfee's* request to discuss a merger between our two companies. However, we believe that the transaction proposed by *McAfee* is not in the best interest of *Cheyenne's* shareholders.

'A transaction between *McAfee* and *Cheyenne* would likely be highly dilutive to *Cheyenne* shareholders, and its value would be dependent upon *McAfee's* ability to continue growth rates in its primary business – anti-virus software – at their historical pace ... We are skeptical of *McAfee's* ability to maintain its current lofty valuation.'

Huai said further: 'In rejecting the *McAfee* proposal, *Cheyenne's* Board of Directors was advised by *Broadview Associates LP*, *Cheyenne's* investment banker, that the implied exchange ratio resulting from *McAfee's* $27.50 stock-for-stock valuation is inadequate, from a financial point of view, to *Cheyenne* shareholders ...

'While we are committed to examining any and every option that will provide value to our shareholders, we will not allow *Cheyenne* to be snapped up by an opportunistic would-be predator at a discount to its true long-term value.'

Cheyenne sees the timing of the bid as an attempt to exploit recent *Cheyenne* stock prices. Huai said: 'The valuation proposed by *McAfee* also fails to take into account the long-term strengths of *Cheyenne*.'

McAfee has taken over four other companies in the last two years [*see also End Notes and News, p.24*]. For information on *Cheyenne*, Tel +1 516 465 4000, or visit its Web site at http://www.cheyenne.com/. *McAfee* can be contacted on Tel +1 408 988 3832, or on the Web: http://www.mcafee.com/ ▮

Outlaws Revisited

Mark Ludwig's now infamous *American Eagle Publications* has launched an updated version of their 'Outlaws of the Wild West' CD. The new CD is said by its marketing blurb to contain 'nearly three times as much information as the first release'. The CD is also said to contain electronic editions of back issues of such *American Eagle* publications as *CVDQ* and *The Underground Technology Review*.

Virus Bulletin hopes to have more information on this new CD in a forthcoming issue, but urges readers not to buy this or any similar virus collections, for any purpose ▮

Prevalence Table – March 1996

Virus	Type	Incidents	Reports
Concept	Macro	91	20.6%
Form.A	Boot	44	10.0%
AntiEXE.A	Boot	41	9.3%
Parity_Boot.B	Boot	30	6.8%
AntiCMOS.A	Boot	27	6.1%
Empire.Monkey.B	Boot	21	4.8%
Junkie	Multi	18	4.1%
Ripper	Boot	15	3.4%
NYB	Boot	12	2.7%
EXEBug	Boot	10	2.3%
Sampo	Boot	10	2.3%
Stoned.Angelina	Boot	10	2.3%
Telefonica	Multi	10	2.3%
WelcomB	Boot	10	2.3%
Manzon	File	9	2.0%
V-Sign	Boot	6	1.4%
Jumper.B	Boot	5	1.1%
Empire.Monkey.A	Boot	4	0.9%
Stoned.NoInt	Boot	4	0.9%
Unashamed	Boot	4	0.9%
Natas.4744	Multi	3	0.7%
Peter	Boot	3	0.7%
Russian_Flag	Boot	3	0.7%
Stealth_Boot.C	Boot	3	0.7%
AntiCMOS.Lixi	Boot	2	0.5%
Byway.A	Link	2	0.5%
Da'Boys	Boot	2	0.5%
Frodo.Frodo.A	File	2	0.5%
Quandary	Boot	2	0.5%
She_Has	Boot	2	0.5%
Stoned.Kiev	Boot	2	0.5%
Taipan.438	File	2	0.5%
Win.Tentacle	File	2	0.5%
Other [1]		31	7.0%
Total		442	100.0%

[1] The Prevalence Table also includes one report of each of the following viruses: Anthrax, Boot.437, BootEXE.451, Burglar, Cascade.1701.a, Cruel, Diablo, Disk_Killer, DiskWasher, DMV, FITW, Floss, Form.B, Helloween, IntAA, J&M, Ken+Desmond, MtE:Coffeshop, Overboot, Peacekeeper, Phx, Quicky.1376, Screaming_Fist.II.696, Screaming_Fist.650, SF2, Stoned.Stonehenge, Stoned.Swedish_Disaster, Stoned.W-Boot.A, Trojector.1463, Urkel, Yankee_Doodle.TP.44.A.

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 21 April 1996. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

- **C** Infects COM files
- **D** Infects DOS Boot Sector (logical sector 0 on disk)
- **E** Infects EXE files
- **L** Link virus
- **M** Infects Master Boot Sector (Track 0, Head 0, Sector 1)
- **N** Not memory-resident
- **P** Companion virus
- **R** Memory-resident after infection

4Seasons.1514
CR: A stealth, prepending, 1514-byte virus which contains the plain-text strings: '*.dat', 'chklist.cps', 'COMMAND' and the encrypted text: '* THE FOUR SEASONS VIRUS * (C) WET, PARIS 1991 * I HAD MUCH FUN WRITING THIS VIRUS, I HOPE YOU HAVE FUN WITH IT TOO!! * MES AMITIES A PATRICIA M., JE T'EMBRASSE TRES FORT ET JE PENSE A TOI *'.
```
4Seasons.1514       B877 67CD 213D 7386 7478 E8DE 03A1 0F06 80FC 0475 10B4 00B3
```

Baby.116
PR: A 116-byte virus residing in low memory. It contains the plain-text strings 'COCC' and 'EXCC'.
```
Baby.116            B43C CD95 8BD8 1EB9 7400 B440 33ED 8EDD BAE0 01CD 95B4 3ECD
```

Clonewar.551
P: A 551-byte virus which creates hidden, read-only files and contains the text: 'Beyond The rim of the star-light My love Is wand'ring in star-flight I know He'll find in star-clustered reaches Love Strange love a star woman teaches. I know His journey ends never His star trek Will go on forever. But tell him While he wanders his starry sea Remember, remember me.' and '[TrekWar] *.EXE'.
```
Clonewar.551        B43C CD21 723A 93B9 2702 BA00 01B4 40CD 21B4 3ECD 21BA 5B02
```

Detic.1514
CER: An encrypted, appending, 1514-byte virus which contains the text 'C:\COMMAND.COM', 'C:\(_Free_D.)' (the name of a created directory), and '[Friends] Virus V1.00 Virus Deticadet To My ExFriends. Virus Written By [_Free_D.] Made In ALBANIA.'.
```
Detic.1514          B9AF 058B DE50 03F1 2E8A 4701 2E30 0743 E2F6 582E 3004 EB12
```

Hickup.1867
CER: A polymorphic, appending, 1867-byte virus which infects COM files only if they begin with a 'JMP' instruction (E9h). It contains the string 'V3HWPTVTBAVISACN'. The virus code includes a procedure which formats the first hard disk.
```
Hickup.1867         8CC8 8ED8 8C84 7500 8EC0 83C6 7790 8BFF 8BFE B9D4 06FC AC34
```

HLLO.OJ.15788
EN: An overwriting, 15788-byte virus containing the text: 'O.J. Simpson in Guilty!' and 'cd\ * *.exe cd\ cd\ *.exe rb+ rb wb %s ab rb rb'. A reliable search pattern for this virus is non-trivial.

Hole.476
CR: A stealth, appending, 476-byte virus which resides in the Interrupt Vector Table. All infected files have their time-stamps set to 62 seconds, and every file has the text 'Asshole' located at the end of code.
```
Hole.476            B43F CD8B 8BF2 8B04 32C4 3C17 740B B800 57CD 8B83 F11F F6C1
```

Hue.482
CR: An appending, 482-byte virus which contains the plain-text strings: 'I am developing !!!' and 'Tu Hue'. The latter is found at offset 0003h in all infected files.
```
Hue.482             B4CD CD21 3CDC 746B A102 002D 3F00 A302 008E C08B F583 EE03
```

Ioe.239
CO: An overwriting, 239-byte, direct infector which displays the message: 'Internal opcode error.'
```
Ioe.239             B440 B9EF 0181 E900 01BA 0001 CD21 B43E CD21 4783 FF0F 75CD
```

Koufidis.1648
CR: A stealth, encrypted, appending, 1648-byte virus containing the text: 'Koufidis Series (c), Distortion Utilities, Athens 92'. Since the virus keeps its code in memory encrypted, the template below, whilst identifying infected files, does not detect the virus in memory.
```
Koufidis.1648       06EB 1490 2E8A 47FF 83C3 0B90 B94A 062E 2807 43E2 FAC3 ??BB
```

Major.1644
ER: An encrypted, appending, 1644-byte virus which contains the messages: 'The Major BBS Virus created by Major tomwn to DOS)', '\BBSV6\BBSAUDIT.DAT', '\BBSV6\BBSUSR.DAT', 'Puppet', 'Image', 'Gnat', 'Minion', 'Cindy' and 'F'nor'.
```
Major.1644          028B C32B C603 F08B CA8B FB81 C730 0088 0D43 81FB 3B06 75DB
```

Mand.1061
CER: A stealth, appending, 1061-byte virus which avoids infecting files with the string '*MAND????' in their names, and EXE files with byte at offset 0Ah set to zero.
```
Mand.1061           C745 0352 00B4 F3CD 21E3 2856 06C6 4501 0841 8EC1 0E1F B911
```

NRLG.968
CR: A stealth, encrypted, 968-byte variant. It does not hide its presence in files of less than 1000 bytes.
```
NRLG.968            E800 008B FC36 8B2D 81ED 0301 2E80 3E41 01B9 743B B9C8 048D
```

NRLG.990

CR: A stealth, encrypted, 990-byte virus containing the text: '[NuKE] N.R.L.G. AZRAEL'.

```
NRLG.990          E800 008B FC36 8B2D 81ED 0301 2E80 3E4F 01B9 7449 B9DE 048D
```

Nado.838

CR: A stealth, 838-byte variant [*see VB, April 1996*] which contains the text: 'anti-vir.dat' and '[Yitzak-Rabin 1.00 (c) made by TorNado in Denmark'96]'. The stealth routine has the same bug as the original – when the virus is active in memory, some clean files appear to be 838 bytes shorter.

```
Nado.838          3E8B 961E 038D B609 00B9 6401 3114 4646 E2FA C3E8 0000 5D81
```

One_Half.3518

CEMR: A stealth, multi-partite, 3518-byte variant containing the text: 'A20 Error !!! Press any key to continue ...' and '.COM.EXE SCAN CLEAN FINDVIRU GUARD NOD VSAFE MSAV CHKDSK'.

```
One_Half.3518     B859 5115 56C2 72F9 D4FF 88E0 8ED8 89C3 80CB 06FF 77FE FF37
```

PSMPC.227

CN: An appending, 227-byte direct infector; infects three files at a time; contains the string: '*.com'.

```
PSMPC.227         B002 E852 00B4 40B9 E300 8D96 0301 CD21 B801 572E 8B8E FE01
```

PSMPC.548

CEN: An encrypted, appending, 548-byte virus containing the text: '[MPC]', '[Skeleton]', 'Deke' '*.exe' and '*.com'. All infected EXE files are marked with 'AD' at offset 0010h. The virus uses two slightly different encryption schemes:

```
PSMPC.548         BF0A 01BE ???? 2E81 04?? ??46 464F 75F6
PSMPC.548         BF0A 01BE ???? 2E81 2C?? ??46 464F 75F6
```

PSMPC.808

EN: An appending, 808-byte virus which marks all infected files with 'PH' at offset 0010h from the beginning of the file header. Because of a bug in its code, the virus reinfects already infected programs. Its code contains a procedure to overwrite the hard disk, and includes the text: '/\ ///\\\\ ///////\\\\\\\ NYC GEN 1 by CrAzY NuTz PEaCE To Da JizzA'.

```
PSMPC.808         A904 5048 5A58 5BE8 4E00 0528 0383 D200 B109 50D3 E8D3 CAF9
```

Rainbow.2337

CEDMR: A multi-partite, 2337-byte variant of the Rainbow.2351 virus [*see VB September 1995*]. It contains the same strings: 'HiAnMiT - roy g biv' and '*4U2NV*'. The following template detects infected files and the virus active in memory.

```
Rainbow.2337      E800 005E 83EE 03B8 AD1B CD13 3DED DE75 450E 1F81 C65F 0781
```

Raveica.680

ER: An appending, 680-byte virus displays the text ' Ha!Ha!!Ha!!! You Have The Raveica Virus V1.3!' on 30 August. It contains a procedure to overwrite the hard disk.

```
Raveica.680       891E AB02 8C06 AD02 BA80 00B8 2125 CD21 0E1F 8CCB 3E2B 9EA6
```

Raveica.764

ER: An appending, 764-byte virus which contains the text displayed on 30 August: ' Ha!Ha!!Ha!!! Ai un virus! Pt. obtinerea devirusorului grabiti-va sa-l felicitati astazi pe Claudiu Raveica cu ocazia zilei de nastere Adresa:Str:Marasesti Bl:11 App:15 Oras:Bacau Jud:Bacau Cod:5500'. The virus contains another message, located at the end of all infected files: 'Bing cu bang'.

```
Raveica.764       891E FF02 8C06 0103 BA7F 00B8 2125 CD21 0E1F 8CCB 3E2B 9EEB
```

SillyC.302

CN: An appending, 302-byte direct infector which infects three files at a time. It contains the text '*.COM','????????COM' and 'GB1.4'. The virus is detected by the following template, but also by the string published in *VB* (August 1992) for the Ash virus.

```
SillyC.302        8D96 0801 B92A 01B4 40CD 21B8 0042 9933 C9CD 218B 863D 0240
```

Syndrome.1485

CER: A stealth, encrypted, appending, 1485-byte virus containing the text: '[Syndrome virus (c) 1996 by The Nuker]'. It reinfects infected files, creating programs with multiple copies of the virus.

```
Syndrome.1485     E81E 008B 861F 012E 8986 0C01 8DB6 3301 B9CE 022E 8134 ????
```

Tet.409

CN: An appending, 409-byte, direct, fast infector containing the plain-text strings: '*.com' and 'Just booted...'. All infected files are marked with the string '383' located at offset 0003h.

```
Tet.409           750F 807C 0438 7509 807C 0533 7503 EB4A 905B 53B0 02E8 8900
```

THU.890

CN: An appending, 890-byte, direct infector with the plain-text messages: '[THU.Suicidal.Dream.A](c) 1996 The Freak/The Hated UndergroundFrom the hypnotic spectre of wake I screamLocked in the depths of a Suicidal Dream', '.com *.zip anti-vir.dat', 'Bad command or file name', and 'Happy Birthday Freaky!'.

```
THU.890           2E8B 8EF9 032E 8B86 3404 81C1 7D03 3BC1 74BE 2D03 002E 8986
```

TV_Nova.665

CR: An appending, 665-byte virus containing text displayed on the seventh day of every month: 'Virus TV N O V A Extremly a n t i heuristic system Technical infos: All is S H I T Greets go to all virus developing groups in Brno ! Czech republic96'.

```
TV_Nova.665       E800 005E 81EE F601 B800 35CD 218D 9415 02B8 0025 CD21 40B8
```

Vienna.480

CN: A 480-byte direct infector which infects one file at a time. It contains the text: 'These days... 19 nov 1988 - "LENIN"'.

```
Vienna.480        B440 8BFA 2BD1 B9E0 01CD 2173 03EB 3A90 3DE0 0175 34B8 0042
```

Vienna.X.629

CN: 629-byte direct infector. All infected files have the character 'X' at the end of the code. The virus contains the text: '*.COM' and 'PATH='.

```
Vienna.X.629      B975 0290 8BD6 81EA E601 CD21 7220 3D75 0290 751A B800 42B9
```

Voices.1900

CER: A polymorphic, 1900-byte virus. It contains the strings: 'discharge', 'sofia', command.com', you keep this love', 'tuturutki', 'possessed', and 'SUICIDAL TENDENCIES'. It is polymorphic: the following template is the only one possible, but reliable detection requires more advanced techniques.

```
Voices.1900       E800 005B B9DC 0531 ??0D 9043 EB00 E2F7
```

VIRUS ANALYSIS 1

SayNay: Making Itself Heard

Eugene Kaspersky

The new viruses I see every day may be divided into three categories. First are the stupid viruses which are totally uninteresting and have no new ideas (but of course they can be fast infectors, and quite dangerous). Then come the monsters intended to bend the minds of anti-virus gurus: it seems that virus authors spend more time writing and debugging their techno-children than anti-virus vendors spend producing their detection and disinfection routines. The last category contains curious viruses that bring us new ideas – not always dangerous, but different.

Looking at curious viruses is much more rewarding than looking at their destructive or polymorphic brethren. Sadly, the latter appear more often. Unfortunately, the answer to the question 'what's new in the virus field?' is usually 'ten or more very dangerous and polymorphic viruses'.

But the box of curious viruses is not empty, and new ones are sometimes found within – the latest is SayNay, a 5115-byte virus which, whilst not intentionally destructive, is worthy of note as one of its two infection techniques is very unusual.

COM File Infection

There are no surprises in SayNay's main infection routine. The virus infects only COM files, and infected files have a JMP instruction at the beginning: when such a file is executed, the JMP passes control to the virus code.

When SayNay receives control, it gets its offset through a standard method – it performs a CALL instruction, gets the stack pointer and takes the word from the top of the stack.

Then the virus searches for COM files using the DOS functions FindFirst/Next by name (Int 21h, AH=4Eh, 4Fh) with the mask:

```
*.co?
```

and infects matching files in the current directory. During infection, the virus reads nine bytes from the file header by way of self-recognition, looks for the ID-string 'SayNay' starting at the third byte. If this is not found, it writes its code to the end of the file, and overwrites the file header with a JMP VIRUS instruction followed by its string:

```
E9 xx xx "SayNay"
```

Whilst it infects, the virus gets and later restores the file's date and time stamp, and clears, but does not restore, the file attributes. There is nothing interesting, nothing strange, in this. It is just a very simple infection routine which occupies only about 200 bytes.

From the Source's Mouth...

The only point about SayNay which is interesting enough to make me write this article is the fact that it may drop its own source code into an ASM-file. I cannot remember any another virus which has an executable format (COM, EXE, SYS, etc) and can do this as well.

To accomplish this, the virus contains its own encrypted source code (4633 bytes) within its body (this is why the virus is so long).

The source code is only dropped by SayNay if a user asks it to do so. Before passing control to the infection routine, the virus checks the command line arguments given to the host program by the user. If the first argument starts with 'NAY', the virus calls the dropper code.

First, the trigger routine displays a message to a lucky user:

```
Magic! ;)
```

Next it creates two files, called SAYNAY.ASM and SAYNAY.BAT. Then the trigger routine decrypts both its own source code from within the virus body, placing it in the .ASM file, and some batch file commands, which it places in the .BAT file.

When this is completed, SAYNAY.BAT contains the commands:

```
TAsm /M2 SayNay.Asm
TLink /T SayNay.Obj
Copy /B SayNay.Com+SayNay.Asm
```

and SAYNAY.ASM contains the 4633 bytes which make up the virus' source code.

As a result, there are two new files in the current directory. The first contains the virus' source code; the latter, instructions on how to compile the source to build the virus.

When it is executed, the BAT file executes the *Borland* assembler and linker (if these are not present, the batch file will fail) to make 'intermediate' virus code which contains the binary code of the infection and the trigger routines, but not the source text.

Then it appends the source text to binary code by using the COPY command. The resultant file (dropper) contains the virus code along with the source in a non-encrypted form. When executed, this dropper (called SAYNAY.COM) encrypts this source before searching for and infecting any COM files.

The cycle is now complete: the virus has produced its source code and the batch file, the batch file has created the dropper, and the dropper then infects files with the same virus as the original.

Figure 1: Nuclear themes abound in recent viruses. SayNay is the latest of these.

The Text Strings

The virus stores some text strings 'in clear' within its body:

```
SayNay
naysaynay.asm saynay.bat
Magic! ;)
```

All other strings (BAT commands and ASM source code) are encrypted within the virus body. The BAT-commands are described above, and ASM text contains the header shown in Figure 1 above.

Conclusions

SayNay is a curious little virus – it could spread in the real world via its primary infection technique; that of directly infecting COM files. The secondary technique, however, is a different matter. As this is only activated when the user gives a specific command-line argument, in the real world this will not become an issue.

SayNay

Aliases:	None known.
Type:	Non-memory-resident parasitic infector.
Infection:	COM files only.
Self-recognition in Files:	Compares six bytes at offset 3 in the file with the string 'SayNay'.

Hex Pattern in Files:	`FAE8 4F01 3E8B 6E00 81ED 0D01`
	`FB8D B697 02BF 0001 B909 00F3`
	`A4BE 8100 8DBE 6E02`

Trigger:	Displays message, creates ASM and BAT files. See analysis for details.
Removal:	Under clean system conditions identify and replace infected files. Also look for and delete the files SAYNAY.ASM and SAYNAY.BAT.

VIRUS ANALYSIS 2

Winlamer

Igor G Muttik

It has now been some time since the appearance of the first polymorphic virus. These have led, perhaps unsurprisingly, to the development of polymorphic construction sets and engines, which are available as executables, linkable object files, and source code. It is a common occurrence to see a highly polymorphic virus based on an engine, or using its own generator: the very fact of their variability makes reliable detection difficult. Until the appearance of Winlamer, such viruses could infect only normal DOS executables. Why?

The answer to this is twofold. Apart from the obvious reason that it is more difficult for a virus to infect a *Windows* program than a COM file or a boot sector, under *Windows*, program code is write-protected, so a program cannot modify itself, and self-encrypting code cannot exist at all. Therefore, at first sight, it may seem that the existence of a polymorphic virus under *Windows* would be impossible. Winlamer has overcome the problems involved, however, and has thus become the first polymorphic virus for *Windows*.

Execution of the Infected File

Winlamer is a direct-action (non-resident) virus, which infects NE-format programs. NE (New Executable) is the standard format for 16-bit *Windows* applications; almost all *Windows 3.1* executable files use it.

When a Winlamer-infected file is executed, control passes to the decryption routine. Winlamer, like all polymorphic viruses, is encrypted: to get to the virus body, the polymorphic code must first decrypt it. This implies that the virus must be able to modify its own code – under *Windows*, remember, code is write-protected when the program takes control.

To allow it to modify its host program, Winlamer uses a simple and obvious method: it issues an Application Program Interface call (DPMI/Windows API: Int 13h, AX=000Ah), which duplicates the code segment selector to the AX register. Then the virus assigns the obtained value to a data segment selector (MOV DS,AX), meaning that for this data segment, no restriction to modify its contents remains.

The API call is one of the first actions carried out by the polymorphic decryptor: it is concealed by meaningless garbage commands and is issued before the virus has decrypted itself. When it gets write access to the encrypted body, it begins decrypting itself, using a simple 'XOR [BX], KeyByte' instruction. The decryption loop follows the API call in the polymorphic decryptor.

The virus body takes control on decryption. The contents of the DS register are restored, and Winlamer then issues a call to check whether DPMI is loaded (Int 2Fh, AX=1686h).

This might seem strange, as the virus has already used a DPMI service: *Windows 3.1* has a built-in DPMI driver, so under all circumstances the DPMI driver should be available. If the virus detects that DPMI is not responding, control passes to the host file. If the program is run under an artificial environment (debugger/emulator), the virus will not replicate.

Then the virus allocates memory and sets the necessary access rights (again using DPMI services). Its remaining actions are similar to those of normal direct-action DOS viruses: get DTA (Disk Transfer Area), get/save current directory, set the current directory to \WINDOWS, and issue FindFirst/FindNext (Int 21h, AH=4Eh,4Fh) calls until all available victims are infected. Winlamer does not check whether the WINDOWS directory exists, so will not infect if the name of this directory was changed when *Windows* was installed.

The virus seeks and infects all EXE files with the signature 'NE' in the WINDOWS directory, but not those executables with other extensions (e.g. screensavers in SCR files). It tries to infect all NE-format files in the WINDOWS directory in one go – the time it takes to do this is very noticeable. When all victims are infected, control returns to the host file.

Winlamer appends its body to the victim file and sets the necessary entries in the NE-header to give itself control when the program is executed. File size change is variable, but is typically 2000-2100 bytes. Infected files are marked by adding 100 years to the time stamp – this is not visible under DOS (the first two digits of the year are shown neither by the DOS DIR command nor by most other disk utilities).

Virus Internals

Winlamer has no payload. The virus was written by a productive virus author; someone who calls himself 'Burglar' from Taiwan. It carries the following strings:

```
Winlamer2 (C)Copyright Aug, 1995 by Burglar in Taipei.
PME for Windows v0.00 (C) Jul 1995 By Burglar
```

Burglar has written other sophisticated viruses. In June 1995 he wrote Wintiny.741; a non-polymorphic direct-action NE infector. Winlamer's code is very similar to that of Wintiny; the infection routine is nearly identical. In fact, Winlamer could be called a polymorphic variant of Wintiny.

He also wrote the Phantasie Mutation Engine (PME), the generator used in his DOS viruses and in Winlamer. It took only two months for his code to evolve from being able to write a normal direct-action NE infector to polymorphic code. This must have taken a great deal of hard work!

The Polymorphic Engine

Winlamer's polymorphic engine (PME for *Windows*) is based on the PME for DOS, but is much more simple. Like almost all polymorphic engines, it has a random number generator (RNG) – the virus reads three values from port 40h (timer), XORs the last two values, and RCRs the result using the first value as a shift counter. The RNG returns a random value in AL register. The engine uses a set of tables of subroutines, and 'dissolves' the basic decryption routine in the garbage commands – not exactly the highest level of polymorphism.

Most of the garbage commands generated by the engine are one byte long (five such commands are used: NOP, REP, REPNZ, CLD, and STD), though some are two bytes long (XOR, OR, AND, ADD, ADC, SUB, SBB, and CMP performed on two registers) and some four (MOV, OR, ADD, ADC, SUB, SBB, and CMP performed on one register and a constant). Winlamer does not generate any three-byte garbage commands; where these occur in the polymorphic code, they belong to the decryption routine.

The virus uses an elegant method to increment the BX register in the decryption loop (instead of the obvious INC BX). It executes two commands to achieve this: NOT BX and NEG BX. The purpose of these may at first be unclear, especially when separated by the polymorphic garbage which was undoubtedly used to add obscurity to the decryptor.

Conclusion

Winlamer has not yet been reported in the wild. Fortunately, it cannot infect both normal DOS EXE and NE files – if it could, it would be much more virulent. This may be an experimental virus, which would explain why it has no payload and its replication is so noticeable. It may well have been written simply to prove that it is possible to have polymorphic *Windows* viruses. Winlamer looks neat, well-coded and elegant.

We already have Winsurf, a memory-resident *Windows* virus. How much time will it take for virus authors to mix techniques and create resident polymorphic viruses for *Windows*, and viruses infecting DOS executables, *Windows* and *Windows 95* applications? All the necessary approaches have been tested separately. The only thing left is to combine the techniques. And that should not take much time…

Winlamer	
Aliases:	Winlamer2, WIN:Lame.
Type:	Direct action NE-EXE file infector, polymorphic.
Infection:	EXE files (MZ and NE only).
Self-recognition in Files:	Adds 100 years to the file's time-stamp.
Hex Pattern:	The virus is polymorphic: no simple pattern is possible.
Trigger:	None.
Payload:	None.
Removal:	Use backups or reinstall files from original diskettes.

VIRUS ANALYSIS 3

Waving the Flag

Kevin Powis

Russian_Flag is an in-the-wild boot sector infector. It infects fixed and floppy disks and carries a date-based trigger which displays an image of the Russian flag (hence the name).

If an infected disk is left in the floppy drive during booting, the virus is loaded into memory and control passes to it. This applies to all boot sector programs, including viruses, but as the small area of memory where the ROM BIOS places the virus is reserved for the 'real' boot sector, the virus must relocate itself away from here before it lets the PC continue to boot – a standard problem for boot sector viruses.

Loading the Code

Now, they say there is more than one way to skin a cat, but not being into that sort of thing I wouldn't know. There is certainly more than one way to relocate a boot sector virus, and the ingenuity of virus authors never fails to amaze me.

The standard method is to decrement a low memory word that controls the amount of memory the PC thinks it has: the virus relocates itself into the 'missing' space. A 640KB PC would think it had 639KB – the code used to do this can be generically identified by many scanners. To avoid such problems, the author of Russian_Flag is more adventurous.

The first segment of conventional memory (segment zero) is 64KB long. The virus sits about halfway through this at offset 7C00h and needs 328 bytes in which to reside while the PC boots and in which to live while the PC is running. The author of Russian_Flag solves this problem by utilising the space taken by 'unused' vectors in the interrupt table.

The first 1024 bytes in segment 0 contain 256 four-byte pointers called interrupt vectors. Each has a segment:offset pointer to the program code controlling its associated interrupt; e.g. vector 13h (the disk vector) contains a segment and offset pointer to current disk handler code. When an Int 13h instruction is given to the CPU it will multiply 13h by four to find the correct vector. From this vector the address of the disk handler can be found and control passed to this routine.

The author of Russian_Flag assumes that vectors 78h-CAh are unused, which is not always true. They are designated as available for programmers – but I don't think that this is what *Microsoft* had in mind. Russian_Flag copies itself into these vectors: from this point on, if any interrupt in the above range is generated or chained, the PC will hang.

Once Russian_Flag has copied across its image it passes control to the copy of itself at the new location. Next it hooks the disk interrupt vector (13h), which will allow the virus to monitor all disk activity.

Russian_Flag is now installed and needs to allow the PC to boot as normal. It does this by performing a read of the hard disk boot sector followed by an attempted read of any floppy boot sector. The reason for this is that the virus is simulating (in reverse order) the normal boot process. If a floppy is in the drive, its boot sector will be the last in memory. If no floppy is present, that read will fail, leaving the hard disk boot sector image in memory.

Trigger Routine

Before Russian_Flag passes control to the image it has read in, it makes a call to the BIOS Get System Date Routine. If this returns 19 August of any year the virus will trigger; otherwise, control passes to the awaiting legitimate boot sector image which was read in previously and the PC continues to boot as normal.

The 33-byte payload routine consists of three small loops, which display differently-coloured bands on the screen of a colour monitor. This results in the tri-colour display representing the Russian_Flag. This displayed, the virus invokes the BIOS getkey function which will halt the computer until a key is pressed. When a key is pressed, the virus continues and control passes to the legitimate boot sector image, just as on any day other than the trigger date.

Interrupt Handler

The virus' disk interrupt handler is invoked automatically on every disk access for floppy or fixed disks. This allows the virus to provide itself with stealth capabilities and protect itself from being overwritten, as well as to infect other disks.

When any disk activity occurs, Russian_Flag monitors the request: if it relates to a second hard disk it is passed on. If it is a read request, the read is performed, but before allowing the result to be seen, Russian_Flag checks to see if it was the MBR being read. If not, control, and the requested information, returns to the caller.

If it was a boot sector read, offset 41h in the returned data is checked for the value 8ED2h, which indicates an infected disk. In this case, Russian_Flag invokes a stealth routine which retrieves the original boot sector from the place it was hidden at the time of infection: this is passed back to the caller, making the PC appear uninfected.

Infection

If the sector is not infected, Russian_Flag sets about correcting the situation. It calls the stealth routine which calculates the hiding place for the clean sector. The sector is now infected, using the same infection routine for fixed and floppy disks. The first two bytes of the sector are patched with a JMP instruction to offset 40h in the sector and the

virus body copied into that location. The amended image is written to the boot sector and a final read via the stealth routine enables the virus to retrieve and return the original uninfected sector – the target of the original call.

If a write is attempted, Russian_Flag allows it if it is not to the first hard disk. Otherwise, if the target head is not zero the writes are allowed. If the write is to sector 9 (where the original boot sector is hidden on hard disks), Russian_Flag, although returning appropriate register values, does not perform the write. If the write is to the boot sector, the virus amends the target sector to 9 and allows the write. The write then affects the original sector (which is hidden in sector 9) as intended and not the virus image in the boot sector.

The only part of the virus left to describe is the routine which decides where to hide the original boot sector at the time of infection. This uses the value in the DL register to determine whether the target is a floppy or a fixed disk. For fixed disks, head 0, cylinder 0, sector 9 is used, regardless of capacity. For floppy disks, a convoluted algorithm is used which examines values in the floppy BPB and results in head 1, cylinder 0, sector 15 being used on HD 3.5-inch disks and the same head and cylinder but sector 5 on DD 3.5-inch disks.

Summary

Russian_Flag has little to set it apart from numerous other in the wild viruses, other than its technique of using the interrupt vector table in which to hide. Fortunately, the virus writer was satisfied with a visual payload rather than taking the easier option of trashing the disk. The trigger date may be related to that of an attempted coup in Russia.

Russian_Flag

Aliases:	Ekaterinburg.
Type:	Boot sector infector.
Infection:	Floppy and hard disks.
Self-recognition in Boot Sector:	Word value at offset 41h in the boot sector equal to 8ED2h.
Hex Pattern:	On hard/floppy disks and in memory. `80FA 8077 0A80 FC02 742B 80FC 0374 06E8 AB00 CA02 00`
Intercepts:	Int 13h Disk handler.
Trigger:	System date is 19 August any year.
Payload:	Visible only on colour monitors: a tri-colour (white/blue/red) flag.
Removal:	Use FDISK /MBR to remove the virus from a hard disk. Salvage required files (which will be unaffected) from floppy; then format.

COMPARATIVE REVIEW

Macro Malarkey

It has been ten months since the first macro virus made its entrance: Concept [*see VB September 1995 pp.8-9*] now has a place both in history and at the top of the Virus Prevalence Table [*see p.3*]. In the light of this, *VB* decided to examine the state of anti-virus companies' defences against the new breed of macro viruses. Ten months would, one might think, be more than enough time to create effective systems to combat them.

The Problem, The Solution

It is often said that the anti-virus industry is the fastest moving of all the software fields: 150-200 new viruses per month is burdensome by any standards, although very few of those require major work on the part of the vendors.

Macro viruses, however, are completely different from all that came before. The format of the files they infect (*Word* documents) is many times more complex than that of standard DOS executables, and *Word* documents do not have a fixed extension – for total security, adding .DOC and .DOT to the product's checklist is not enough. Equally, the product cannot simply scan every file – the penalty in terms of speed would be unacceptable to users.

Anti-virus companies were asked to provide a product to combat *Word* macro viruses: the various packages received show that the solutions have not yet begun to converge. This will take more time, as some manufacturers replace stopgap solutions with more polished ones, others change direction, and the rest simply enhance their product.

For the moment we have an intriguing mix of scanners, *Word* bolt-ons, macros of various sorts, and combinations of these. They all have advantages and disadvantages, which make a real difference to the level of protection offered by each.

Testing Criteria

Testing for this review is different from the normal tests *VB* performs in its comparatives: the bulk of testing is normally based around scanning both infected and clean files on disk. This review, whilst it does look at the ability of the products concerned to find instances of the viruses in question in static files, also spends time examining on-access checking; that is, whether or not a product could detect the virus in an infected document as it is loaded into *Microsoft Word*.

Immediate detection of macro viruses is vital: documents are passed around both within and between organisations to a much greater extent than diskettes or conventional executable files. In addition, once a computer is infected with a macro virus, it generates infected objects (i.e. *Word* docu-

ments) far more quickly than if it were infected with any other current form of virus. Even if the virus is caught when the machine is next booted at 9.00 o'clock the following morning, in many situations this will be too late to prevent large numbers of documents being infected, with all the problems this entails.

Types of Execution

In DOS, the methods by which a virus can take control are limited. Under *Word*, things are different. A user can open a file in many different ways; for example, by using:

- File/Open to open a document with a known path
- File/Find File/Open to search for, preview, and then open, a document
- File/Insert/File to incorporate a document into the one currently being edited
- Drag and Drop to move a document into the *Word* window, thus opening it
- the Most Recently Used list, at the bottom of the File menu: this allows easy access to the last few documents
- a double click on a document in File Manager: *Windows* uses the association to invoke *Word* automatically
- File/Run in Program Manager to invoke *Word* on a file with a non-DOC extension; e.g. WINWORD C:\TMP\WORDWORD.TMP

These are all different ways, as far as *Word* is concerned, to open a document. In all but one of the above cases, a macro virus in the file being loaded has the chance to gain control and infect the system by copying itself into NORMAL.DOT. The odd man out is File/Insert/File.

Whilst the techniques are different from the point of view of *Word*, if a conventional resident scanner is used, it simply sees a file being opened. Use of a conventional system to find these viruses would seem to be a big advantage. However, on the flip side, it is easier for a *Word*-based solution to determine accurately whether or not a document is truly infected – at this level, access to the macros is that much easier.

Viruses Used

Ten samples each of four macro viruses (Concept, Colors, DMV and Nuclear) were used. The ten files were made up of five which were generated and infected with *Word 6.0*, and five with *Word 7.0* (*Word for Windows 95*, which is fully compatible with *Word 6.0 for Windows 3.1*). Each group of five consisted of one infected copy of NORMAL.DOT, two files infected without Fast Save enabled, and two infected with Fast Save active – Fast Save creates modifications within the file format which can cause problems to some inspection methods.

As it transpired, the different file styles and formats had no effect on all but a very limited number of the products which were tested.

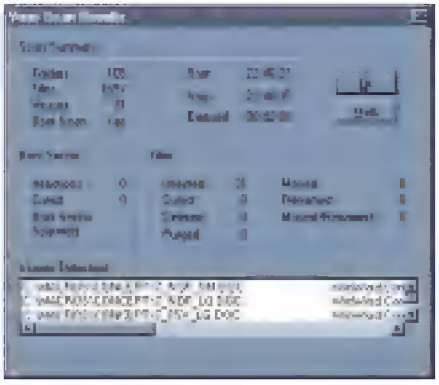
ChekWare ChekWord

This is not so much a virus detector as a macro detector – it hooks File/Open and displays information about macros found in any documents that are opened. The user is offered the chance to remove such macros before opening the document (the original is saved with a different extension in case of errors).

In a situation with expert users who know what is happening, this type of solution may be fine; however, in a more standard situation, where the system is being used by non-experts, more automation is required to solve the problem. In addition, documents opened via menus other than File/Open are never checked.

Cheyenne InocuLAN

Cheyenne submitted its *Windows 95* product for this comparative: its scanner portion found all the virus samples except those of Colors. It was, however, also unable to disinfect two of the ten samples of Nuclear included in these tests.



The resident scanner picked up exactly the same samples, and (as would be expected for a solution of this type) was also able to prevent access to all of the infected files, regardless of how they were opened, as long as the extension of each was .DOC or .DOT.

Command Software F-Prot Professional

The DOS command-line version of *F-Prot* detected all the macro viruses bar one sample of DMV. However, the *Windows* version of the product did not have .DOC and .DOT in its default extension list, so initially missed everything. When the extensions were added, it achieved the same scores as its DOS stablemate. The resident scanner performed identically, and was able to stop the infected files being loaded.

This illustrates an important point, which is that administrators must ensure that their product is actually checking the files in question. They should not assume that all parts of a product will check the same files.

Cybec VET

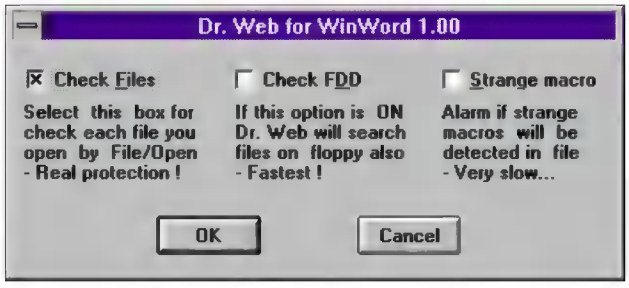
Cybec submitted a macro solution, *VET for Word v2.1*, a document which scans other documents for the virus' presence. When installed, it checks NORMAL.DOT for the presence of Concept, and if found, cleans it, before installing a dummy PayLoad macro to prevent future infection.

The document scanner, like all pure-macro solutions, is slow: the time taken to scan large numbers of documents is prohibitive. It detected all the Concept samples – unsurprising, as this product is designed purely for Concept. One problem is that the default list is to scan .DOC files only: the extension .DOT must be added to the list manually. Also, an error box is shown when disinfecting NORMAL.DOT: *VET for Word* tries to remove Concept's AutoOpen macro, which is not present in infected copies of NORMAL.DOT.

The on-access checker detects Concept just as reliably as the scanner, but only when File/Open is used: all other methods of opening a file bypass the checker. With Concept, this is academic, as the virus will not install in NORMAL.DOT due to the presence of a PayLoad macro, but if and when *VET for Word* is updated to include new macro viruses, this may become a problem.

Dialogue Science DrWeb

This is another macro-only solution: when the document supplied is loaded into *Word*, it installs some macros into NORMAL.DOT. It places an extra menu, called 'Dr Web', onto the word menu bar – this allows easy access to the scanning functions. The scanner finds the Concept and



DMV viruses: it found them twice in each infected file, which was baffling, but it correctly removed the viruses anyway. The on-access scanner had some problems – it closed every file the reviewer tried to open!

When everything was disabled except the heuristic option (*DrWeb* seems to be the only macro solution which currently attempts to analyse macros heuristically), it worked, and found the viruses. The scanning, however, is extremely slow, and renders *Word* close to unusable – the *WordBasic* in which all macros are written is an interpreted language, and as such is not quick.

Eliashim Virusafe

Eliashim provides a separate DOS executable, called VDOC, which is used to scan for any instances of macro viruses. This program was able to find all but one of the virus samples: for some reason, it could not detect one of the instances of Concept.

VDOC also offers the option to disinfect any files it finds infected: this functionality, whilst it works, has something of a problem. It leaves what appears to be the whole virus

behind in the document, so other scanners are wont to suffer ghost positives on the file, believing it to be infected. Strictly speaking, it is not: when such a document is loaded into *Word*, infection does not occur, but it is bad form on the part of *Eliashim* to leave the document in this state. The *Windows 95* resident software was able to detect when infected files (called .DOC or .DOT) were written to the disk.

ESaSS ThunderBYTE

ThunderBYTE 7.01, the version tested here, was able to find all forty infected samples without difficulty (and, as expected, extremely quickly). The reviewer could not, however, make the resident software prevent access to any of the infected files, all of which went on to infect the *Word* environment without difficulty.

In addition, *TBAV* was unable to clean the documents. Indeed, when TbClean (the disinfecting component of the *ThunderBYTE* utility set) is run on an infected file, it attempts to disinfect it as if it were a standard DOS .EXE or .COM file, and corrupts the file.

H+BEDV AVScan

Version 2.65 of this product arrived just in time to be included in the review, and it was able to detect all of the virus samples used. Unfortunately, *AVScan* does not feature (at least in the version supplied for review) any virus removal capabilities at all, and it therefore failed to remove any of the infections.

It is worth noting that the product is designed primarily for a German market, an environment in which Concept does not replicate. That being said, it was the English version of the product which was tested here.

IBM AntiVirus

The version of *IBM AntiVirus* tested was able to detect samples of the macro viruses with the command line scanner; however, it was unable to disinfect them. In addition, the reviewer was unable to configure the resident software to detect the infected files as they were accessed in *Word*, which is a pity, as this is a very necessary part of a defence against such viruses.

Jade ScanVakzin

This product is a command-line or menu-driven DOS program which states that it is designed to clear up macro viruses prevalent in Japan; i.e. Concept. However, it missed the four infections in large document files, making it at best a stopgap solution.

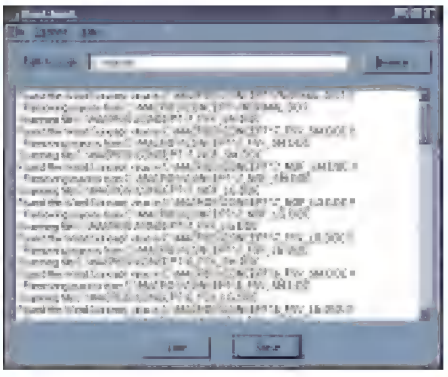
The product does state that it works with Japanese documents – exactly what this means is not clear, but it is a claim that no other manufacturer makes...

Kami AVP

Another macro-based solution, featuring the only document seen in this review with an embedded button. When the button is pressed, *AVP for Word* installs macros into NORMAL.DOT, including one called PayLoad, to prevent Concept installing. The same problems experienced with the other macro solutions appear here: it only hooks File/Open, and macros in files loaded in other ways.

Leprechaun Virus Buster

This solution is one-of-a-kind, at least as far as this review is concerned: it is a *Windows*-only executable which scans *Word* documents for macro viruses. That it is *Windows*-only is significant, as the product uses the *Windows* OLE API to parse the document, thus cutting down the amount of work required for developers. The scanner found Concept and Nuclear, but on-access checking software was not submitted.



Look Software Virus Alert

The DOS scanner submitted by *Look Software* correctly found all forty infected *Word* files, but was not able to clean them. Instead, it recommended deletion, which seems a little harsh for a file infected with Concept. In addition, it seemed to be impossible to make the resident scanner detect access to a file infected with Concept, or to prevent the global template from becoming infected.

McAfee Scan

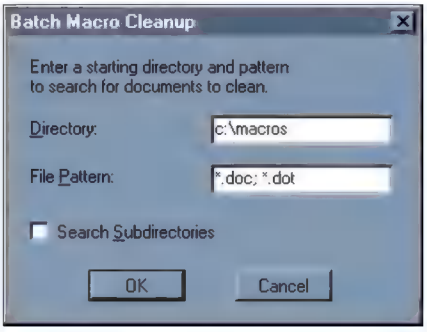
McAfee submitted its DOS and *Windows* products for this test, both of which found all of the macro virus samples with which they were presented. Unfortunately, VShield (*McAfee's* resident utility) did not seem to be able to find any of the viruses, regardless of configuration.

The scanner was able to repair the infections, which it did by wiping the bytes that make up the macros from the file – this level of disinfection is necessary to prevent other products from seeing the virus code and believing (erroneously) that the file is still infected.

Microsoft ScanProt

Famous as the first 'quick fix' to the problem of what *Microsoft* then called the Prank Macro, this solution is specifically aimed at Concept. As expected, therefore, it detected all the samples of Concept, but missed all those of the other viruses.

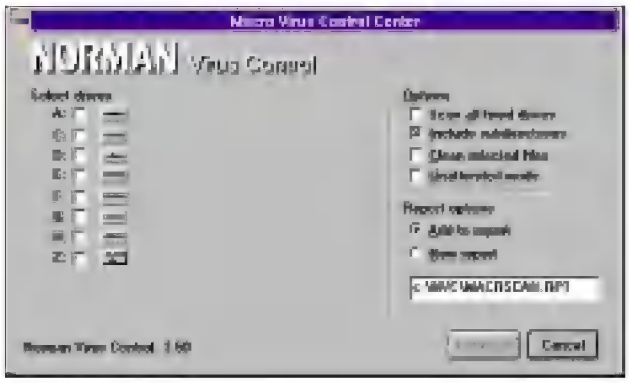
When installed, if it finds Concept in NORMAL.DOT, it will simply rename the macros, leaving their content intact. This causes problems for some scanners, which identify the template as infected. It places a macro named 'PayLoad' in NORMAL.DOT to prevent Concept from infecting.



The document scanner and cleaner is annoyingly slow, and insists on asking at least one question per cleaned file, something which will not endear it to people who have to clean up large numbers of infected documents. In addition, like several other macro solutions, it cannot deal with huge directory structures.

Norman Virus Control

This is an extremely-slick looking defence: whilst the front-end is presented via macros, a DLL provides some core functionality, and the two components combine to provide a very professional interface. Once the product is installed (a procedure which must be performed from the A: drive), it has added to *Word* a floating toolbar called Norman, which may be docked at the top of the screen like all other toolbars. This toolbar contains a button which, when pressed, calls up the scanner: it found all forty infected files, and was able to clean them.

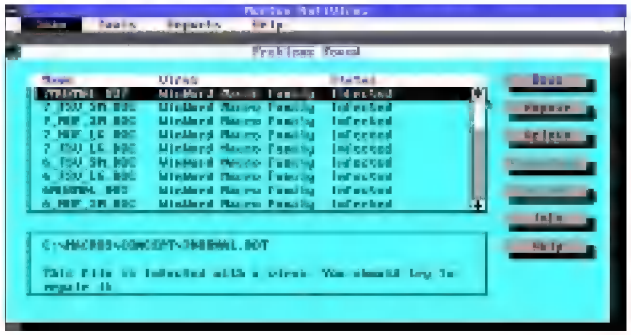


There is a separate system for on-access protection, which is provided by several individual macros. It is clearly designed for corporate environments in which one of the viruses is known to be prevalent in that organisation. These are very automated solutions: when an infected document or template is found, it is disinfected without the user being given a chance to prevent it. This cleaning is done when the infected file is closed, thus neatly dodging many of the effects of opening a document in different ways.

Norton AntiVirus

Norton AntiVirus was one of the comparatively few products able to detect all the macro viruses in the test; however, it could not repair any of the infected files.

In addition, the reviewer could not configure AutoProtect (the resident solution) to prevent macro-infected documents from being accessed, which is something of a shame.



On Technology Macro Virus Track

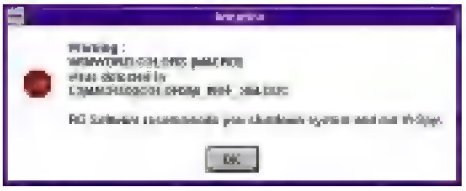
On Technology offers, courtesy of recently-acquired *Thompson Network Software*, the only WLL solution to the macro virus problem. A WLL is a Word Link Library, a method by which programmers may add functionality to *Word* outside the macro environment. Any WLL files in *Word's* STARTUP directory will be loaded and used by *Word* as it starts.

Once it is installed, Virus Track is invoked when *Word* is started (a huge bitmap, which is displayed at this point, whilst very pretty, is a time-consuming pain). The scanner found all forty of the infected samples, and the on-access checker found viruses opened in every way except two: inserting a file, and double clicking on a file in the File Manager.

The product has an interesting approach to the problem of the Most Recently Used list: when *Word* is started, Virus Track immediately scans all the files in the list, and, if viruses are found, removes them – thus it avoids much of the problem. The on-demand scanner found all forty infected files, and cleaned them all without apparent difficulty.

RG Software Vi-Spy

RG Software's on-demand scanner and on-access resident component have both been updated to detect macro viruses; a task which they perform very well. All forty samples were correctly detected by both the scanner and the resident software, but only the ten samples of Concept could be disinfected. For the other files, deletion was recommended.



S&S Dr Solomon's AVTK

S&S has, like *Cheyenne* and *RG Software*, modified both its on-demand scanner and its resident software to detect macro viruses – and to good effect. The scanner was able to detect

and disinfect all forty of the test samples. The macros were completely removed from the file, and only the names were visible after disinfection: no other product falsely found them to be infected.

The resident software correctly detected and notified the user of any access to an infected sample, provided it possessed either a .DOC or .DOT extension (detecting files with other extensions required the VxD to be reconfigured). Overall, an impressive performance.

Sophos Sweep

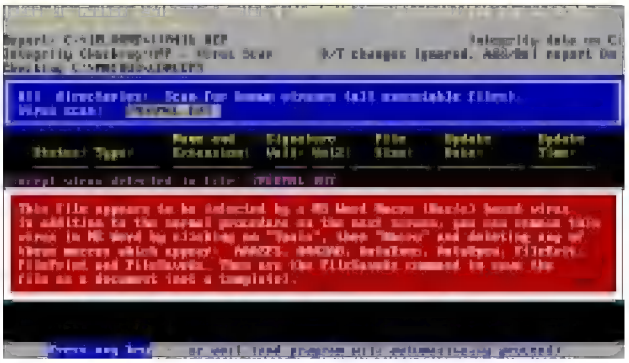
A fascinating two-pronged approach from *Sophos*: their scanner (*Sweep*) and a macro system (*Sword*) are combined to provide a macro virus cleaner. *Sword* parses *Sweep's* log file to discover which files are infected, and then uses *Word* functionality (in the same way as macro-only solutions) to remove the macros. The combination worked well in this test: it correctly detected and removed all the viruses.

On-access protection is not provided by *Sword*; rather, by the resident protection component of InterCheck: this intercepted the opening of the .DOC and .DOT files, and sent them to the server for checking. When they were found infected, the File Open was not allowed to proceed.

This procedure is successful in all situations in which the standard VxDs and TSRs work; that is, those where the files are named .DOT or .DOC. To catch files with other extensions, the InterCheck resident software, like all the others of its type, needs to be reconfigured with a new extension list.

Stiller Research Integrity Master

Integrity Master's scanning component knows about the Concept and Nuclear viruses, but in this test it failed to detect two samples of each of these viruses. When it detects an infection, the user is informed how to use *Word* to remove the viruses: this is fine for a small-scale infection of the two viruses mentioned above; however, with other viruses (for example, Colors), or large numbers of infected objects, a more automated solution is preferable.



Conclusions

As was expected when these tests were undertaken, there are currently numerous different solutions to the problem of detection and removal of macro viruses. Many of these have their own benefits; equally, their own drawbacks.

Overall, however, it does appear that the technique of simply enhancing the product's conventional scanner and memory resident utility offers the most benefits. First, there is no problem with *Word's* multiple methods of opening a file; the resident package is sitting beneath *Word*, and watching for file accesses at the operating system level. Second, there is less need to worry about active macro viruses, for much the same reason – the protection is not dependent on the fact that the *Word* environment has not already been subverted.

On the other hand, from the programmer's point of view, writing macro scanners and cleaners in the *Word* environment is considerably easier: instead of having to update a conventional DOS scanner to understand the vastly complicated *Word*/OLE file format, *Word* (or *Windows*) can do the work, leaving the user with a much simpler task. It is easy to see, therefore, why many of the solutions are at this stage.

The best solutions tested, in terms of all-round automation and removal ability, were *Dr Solomon's AVTK* (especially notable for good removal and memory-resident features), *Norman* (best *Word*-based solution, and *OnTrack* (an interesting technique, especially for dealing with the problem of the Most Recently Used menu options).

Closing Thoughts

Whichever anti-virus product your company uses, bear in mind that if you use *Word* anywhere, the chances that you will encounter Concept are far from insignificant. Make sure that you have systems in place now to prevent it becoming widespread in your organisation: it will save a great deal of time and trouble later on.

Technical Details

Hardware used:

Compaq ProLinea 590, 16MB RAM, 2.1GB disk and a 270MB SyQuest removable drive.

Software:

MS-DOS 6.22, Windows 3.1 and Word 6.0; or (in some cases) Windows 95 and Word 7.0.

Virus samples:

Ten samples of Colors, ten samples of Concept, ten samples of DMV, and ten samples of Nuclear; in a variety of document types and sizes.

Other Technical Information:

After reviewing each product, a complete disk image of the relevant operating system, applications, and samples was restored to the server from a sector-level backup on the SyQuest drive, and the next product was then installed.

FEATURE

Wacky Widgets, Wacky Costs: False Positives

Christine M Trently

As statistics show, computer viruses are more prevalent than ever. The paranoia surrounding infections is increasing, and when an anti-virus product reports an infection, people notice. The anti-virus product false alarm rate is a critical factor during product evaluation and selection.

This study, based on events at a real company (all names have been changed), aims to give a perspective on the cost of false alarms, to draw conclusions about their effects, and to provide insight into reducing the impact of false positives.

The Incident

The *Wacky Widgets Corporation* is an organization of 3,000 to 6,000 employees with sites world-wide. The Research and Development (R&D) department has 25 employees with a wide range of technical expertise who build better widgets using the latest technology. The company has anti-virus policies and procedures implemented, and uses the anti-virus product *XYZ* to provide protection for company workstations.

Soon after the R&D department recently upgraded the operating system on several workstations, *Wacky Widgets* notified all employees of an upgrade to its anti-virus product, which employees were encouraged to load and run.

At 9.00 a.m. on Day 1, Jane loaded the update and checked her workstation. She received a report that a file on the workstation was infected with a polymorphic virus which *XYZ* could not remove. Jane immediately called the technical support group for assistance, and was told to remove her workstation from the network so the virus could not spread, and send them a copy of the file on diskette. Jane did this, and then reported the incident to her manager and co-workers.

Early next day, Jane contacted the technical support group to determine what should be done to contain the virus and return her workstation to an operational state, believing the anti-virus software, *XYZ*, had reported a real infection. The technical support stated that the infected file should be removed and a clean copy used to replace it, though they had little information about the virus.

The reportedly infected file was a critical system file. To avoid major re-installation and restoration delays, Jane decided to replace the file with a clean copy from another workstation. By 9.00 a.m. on Day 3, Jane had checked all compatible workstations in the department: each reported the same infection. Technical support instructed the department to remove 'infected' workstations from the network, to remove the infected file, and to attempt to re-install the operating systems from diskette. This would help determine if the diskettes were infected.

Jane diligently checked other workstations in the department by 2.00 p.m. on Day 3: only those with the upgraded OS reported a problem. Now the site had seven reportedly 'infected' workstations, with no resolution.

At 10.00 a.m. on Day 4, additional technical support was brought in to research the virus and to check the OS installation diskettes. The diskettes were checked for viruses: none were reported. However, system files on the installation set were compressed and *XYZ* could not check them. Technical support remained unable to determine the cause of the infection or the resolution, but was convinced it was real.

Around noon on Day 7, technical support returned with the news that the vendor of *XYZ* stated that the system file was not infected; that the product had reported a false alarm. It had taken six working days for the company to ascertain this.

Time Lost

In calculating the time lost due to the incident, time spent checking the original workstation and researching the virus is not included, as this would have been necessary if the infection were real. Time spent on activities that would not have been necessary without the false alarm has been added, including time spent checking other departmental workstations and OS installation diskettes, and time spent coordinating, waiting for network connectivity, and discussing the incident.

Such information provides a basis for attempting to calculate the total productive time lost. While the lost time would be similar for a real infection, the times identified in Table 1 were not lost due to a computer virus, but to a false alarm.

The time it took to check all workstations in the department is included in productive time lost, because it kept Jane from performing regular duties. In addition, checking the workstation detained the workstation user while checking was being done and after infection was reported.

Checking each of the twenty-four workstations involved with *XYZ* took circa fifteen minutes; a total of six hours to check all workstations in the department. Users of 'infected' computers were detained at least another fifteen minutes to determine possible sources. Six PCs were identified as 'infected', a total of six hours. The total time spent checking workstations was seven hours.

It took approximately one hour to check all OS installation diskettes. This length of time was minor in the scope of the incident but is included, as it was an activity that took an employee from regular duties.

Coordinating the incident included systematic checking of other compatible workstations, interviewing and discussing the infection with workstation users, taking notes on events as they unfolded, reporting results to technical support staff, and completing/filing incident reports to management. This took about six hours over the course of the incident from the initial report to its closure.

During the incident, the site followed the recommended virus containment procedures by removing 'infected' workstations from the network. Workstation usage was thus limited: this had a profound effect on employees who used PCs to receive and respond to electronic mail messages, printing documents through print servers, utilizing the Internet, etc.

Jane, who reported the 'infection', was removed from the network for the full six days, and remaining employees were removed for at least four full days. This makes a total of 244 potential hours waiting for network connectivity. The number of potential hours lost does not include the impact on workers within *Wacky Widgets*, or on clients dependent on connectivity with the 'infected' department.

Activities	Hours lost	Costs (US$)
Checking workstations	7.5	637.50
Checking installation diskettes	1.0	85.00
Coordinating the effort	6.0	510.00
Awaiting network connectivity	244.0	20,740.00
Discussions etc	17.5	1487.50
Total	**276.0**	**23,460.00**

Table 1: Calculations of the total amount of productive time lost through, and costs involved in, this false positive report.

As each workstation was checked and an infection reported, employees were caught up in the moment and long discussions ensued (e.g. how the virus got there, how viruses work, lessons learned, policy and procedure concerns). Conservatively, such conversations lasted about 30 minutes for each of the seven employees, adding up to three hours per day. Total productive time lost to discussions was approximately seventeen hours.

Costs

Since the 'infected' department was comprised primarily of senior level staff, the cost is based on the salary of the senior level staff at *Wacky Widgets*. This cost includes benefits, inflation, etc. For *Wacky Widgets*, the approximate cost for senior level staff is US$85 per hour. The information in Table 1 details labour costs associated with time lost.

While the costs associated with this false alarm are large, despite only including productive time lost for employees directly affected, it is easy to see that it could have been more. *Wacky Widgets* had implemented anti-virus policies and procedures, and employees affected were technically competent. Imagine the costs to a company unprepared to handle a computer virus incident, or whose affected employees were barely computer literate.

In 1994, *IBM* estimated that the cost to combat computer viruses within an organization at $300 per PC. For the site described in this case study, that would be equivalent to $7500. One false alarm cost *Wacky Widgets* more than three times the estimated amount to combat computer viruses. In this case, the diagnosis cost more than the cure.

Conclusions

The costs associated with this false alarm should act as a warning for anti-virus developers and vendors to ensure products neither add to nor exceed the costs of combating viruses. Testing and comparing anti-virus products is done using virus collections; thus, emphasis on reducing false alarms is not apparent. Corporations and other organizations will not expend more in time and resources for an anti-virus product which costs more in lost productive time due to false alarms.

The slow response of the centralized technical support and the reliance on one product contributed to the large amount of productive time lost. The time spent waiting to identify the cause of and resolution to the 'infection' was the single most important factor in this calculation.

Wacky Widgets had defined and implemented computer virus policy and procedures which were easy to understand and follow. The company had also implemented and utilized a centralized help desk. These activities helped reduce risks and costs associated with infections as well as with false alarms. To reduce the risks and costs associated with both real and false computer virus alarms, organizations should:

- enhance the centralized help desk's ability to respond in a timely fashion with adequate training and resources
- establish procedures for software distribution which ensure checking for computer viruses before distribution is permitted
- establish a capability to determine virus presence through expert personnel or contractual support
- select anti-virus products with low false alarm rates using comprehensive product evaluations
- verify identification of computer virus infections using multiple anti-virus products
- establish vendor support for technical issues
- protest when dealing with slow/unresponsive vendors.

The costs associated with combating and containing a false alarm are similar to those associated with a real infection. However, from a corporate perspective, there is a great difference between spending time and resources fighting a fictitious battle and spending the same amount of time and resources preventing/fighting an actual battle. [*This article first appeared in InfoSecurity News, issue March/April 1996*]

PRODUCT REVIEW 1

LANDesk Virus Protect v3.0

Martyn Perry

In previous reviews, I have commented on the lack of network management facilities in some server-based products. *Intel's LANDesk Virus Protect 3.0* (*LDVP*) takes us to the other end of the spectrum, offering everything needed for network management of a virus scanner. It was previously reviewed in *VB* in April 1994: how did it fare this time?

Presentation and Installation

The product is licensed on a per-server basis. As well as server support, the licence allows for installation of the workstation version on any clients directly attached to the server, any stand-alone workstations at the same location as the server, and any portable or home computer of employees who work at the same location as the server.

To ensure this facility is not abused, *Intel* reserves the right to conduct audits to verify compliance with the agreement. *LDVP* comes on CD-ROM, along with a licence diskette and a *Macintosh* scanner: *Mac* support is also provided.

The documentation comprises a User's Guide and an Addendum Manual. The main manual covers installation, configuration and execution of the virus protection options for the server and workstations, and reporting and alerting options. It also contains a useful troubleshooting guide and a short glossary.

The addendum contains corrections to the main manual and useful notes on key configuration files and explanation of their contents. There is also a warning about not using VPRULE.COM (Virus behaviour trap – see later) under *Windows 95* – it was still under test at the time of review.

With regard to installation, a word of caution: it should not be attempted as a late-Friday-afternoon-wind-down-for-the-weekend activity, as file and environment requirements must be checked before the software can be loaded.

The usual CLIB version (G or greater) must be present, as well as the *Novell* workstation shells in matching sets. As the software uses *NetWare* communication, specific files for *Netware* under *Windows* support need to be checked, and the correct versions installed.

Inconsistency at this point could result in *Windows* failing to load correctly with *NetWare* drivers present. Moreover, Alert Management Services (AMS) needs Btrieve 6.10c, and increases many of the default environment parameters (e.g. open files, locks etc.) threefold. A copy of Btrieve 6.10c is conveniently shipped on the CD-ROM, but other files must be downloaded.

Installation is relatively straightforward, and well handled by the installation program. A number of files are modified during the set-up process. Most are saved with the extension .AA#, where AA is the same as the original extension and # is a number incremented when necessary, to avoid clashes.

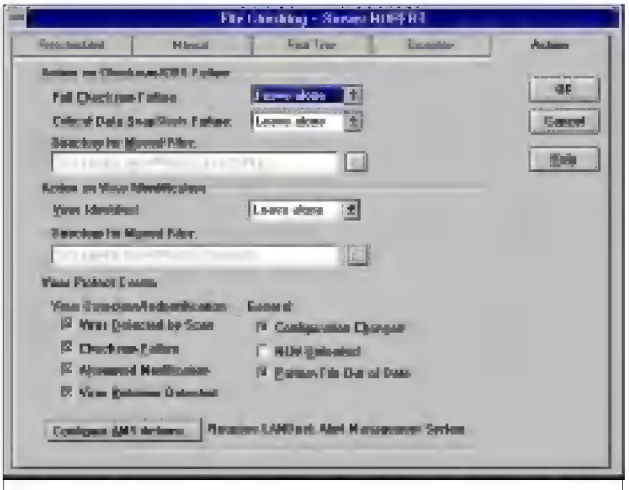
Certain key system files are backed up using a three-digit, progressive numeric extension starting at .001. These key files (SYSTEM.INI, AUTOEXEC.BAT, AUTOEXEC.NCF, NET$LOG.DAT, and WIN.INI) are used for automatic loading of the software on start-up of both the server and the workstations. The software gives the installer the chance to ignore these changes and apply them manually as needed.

The software organises servers into security domains. Each has a main server, controlled by the Management Workstation, and subsidiary servers. The Management Workstation holds various communication support files in addition to the management program.

At the end of the installation, the text file TO_DO_#.TXT lists any outstanding actions to be completed; e.g. files not updated automatically during installation. If there is a problem with installation, the uninstall option uses the backed up files (.AA#) to restore server and workstation to pre-installation state. This appears to work well, which is reassuring, considering the number of files modified or replaced on both server and workstation.

LANDesk Virus Protect

The product employs a number of anti-virus strategies for detection and protection: checking of new files using a combination of pattern scan with known virus strings, full checksum on executable files, and critical data snapshots of key portions of files which are monitored for changes.



One of the pages on the File Checking dialog: the various actions to be taken when a file is found infected may be configured here.

Added protection is provided using an 'Integrity Shield', which protects selected files from modification. It is also possible to create selected Restricted Users, who can be prevented from making modifications to protected files, or to whom file access may be granted for a defined period. Whilst not perhaps immediately obvious as an anti-virus measure, this is very useful from the point of view of a network administrator.

The *LDVP* program is loaded from the server console prompt using VP_AUTO.NCF. This also loads the Alert Management Service, *LANDesk* database manager, Btrieve, and connection management software. Software configuration is performed from the *Windows*-based management workstation. This includes configuring the various modes of scanner operation and resulting actions, and managing alert and report facilities.

LDVP has three modes of scanner operation: Manual, Real Time and Prescheduled. A manual scan will scan the server on demand, using the current Manual settings. Scanning on the server can be started and stopped from the Manual menu on the Management Workstation, or from the server console.

On the test system, when a virus was discovered, the software beeped every time it found a virus – useful in real life, but when testing for over 6000 viruses, it can be too much of a good thing. Information on how to disable this could not be found, but Blu-Tack on the server speaker provided relief, and saved the reviewer being lynched.

The real-time scanner allows incoming and outgoing files to be checked as they are accessed by users. This too can make use of pattern scanning and the virus behaviour monitor.

The prescheduled scan feature allows the server to be scanned on a timed basis (daily, weekly, monthly), using the prescheduled configuration set. There is no hourly timed scan option to allow administrators to provide a regular scan of a specific part of a server during the normal working day.

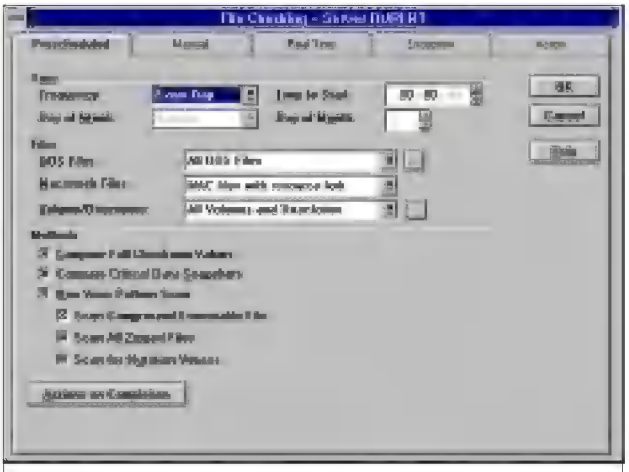
Configuration Options

For each mode of operation, various selections can be made: file extensions to be included in the scan (the defaults are BIN, COM, DSK, EXE, LAN, NAM, NLM, OVL, SYS, VAP, and VIR), volumes, directories or files to be excluded from the scan (no defaults are defined), action on finding a virus (which include: to notify users in a user list, to rename an infected file, to move an infected file to a directory or to leave it alone – the default quarantine directory is SYS:\VPROTECT\VIRUS).

As it is not possible to have more than one prescheduled scan active at any one time, the administrator must change the configuration to whichever prescheduled scan is required.

Workstation Support

Workstation protection is provided using WINProtect. This comprises an on-demand pattern scanner, which uses the same pattern file as the NLM, and a TSR, VPRULE.COM,



Another of the File Checking pages: the Prescheduled option allows the administrator to configure when and what to scan.

which is described as a virus behaviour trap for monitoring for the presence of virus-like activity on local drives. Separate support is included for OS/2 workstations.

Scan reports are created which can be added to the report file on the server at login time so that the status of each workstation can be monitored centrally. *Intel* ships a test file, TESTVRS.COM, which is not a virus but allows testing of report and alert configurations.

Administration

The extra management facilities can be summarised under three groups: report generation, managing alerts and updating server and workstation scanners.

The report generator, ReportTool, is a *Windows*-based program which uses data from *LDVP* log files: this must be converted into comma-separated value (CSV) files before using the Report Manager. Report layout can be formatted to individual needs, or standard templates may be used. Log files can also be viewed directly without exporting to ReportTool.

LANDesk Alert Management System (AMS) provides alert facilities in response to a *Virus Protect* event. These provide various methods of warning users of a problem: a *Novell* broadcast to selected logged-in users, an email message to *cc:Mail/MS Mail/Novell* MHS, a fax, a pager service, an SNMP alarm, run another NLM, a warning in a message box, or play .WAV files (at last, a virus scanner that will whistle Dixie!).

The virus pattern file can be updated by downloading the new file from *Intel's* BBS, manually or automatically, once a month when the system administrator logs on to the main domain server.

The main server then broadcasts to the other servers in the domain that a new pattern has been received, and the servers then attach to the main server and copy the new file. This pattern-sharing can be configured so that all or only designated servers are automatically updated.

Detection Rates

The scanner was run against the usual three test-sets: In the Wild, Standard, and Polymorphic. The undetected viruses were identified by scanning with the 'move to quarantine directory' option and listing the files left in the virus directories.

The Standard and In the Wild test-sets produced a reasonable 95.9% and 94.4% respectively. The Polymorphic test scored 69.2%, due mainly to MTZ.4510 not being detected at all and to the fact that only 339 out of 500 samples of Nightfall.4559.B were detected.

Real-time Scanning Overhead

To determine the scanner's impact on the server, 63 files totalling 4,641,722 bytes (EXE files from SYS:PUBLIC) were copied from one server directory to another. The directories used for the source and target were excluded from the scan to avoid the risk of a file being scanned while waiting to be copied.

Because of the different processes which occur within the server, the time tests were run ten times for each setting and an average taken. The test was performed under eight conditions, including three with real-time scanning on and three with it off. The summary table at right lists the configuration for each test, the purpose of each of which is described below.

The time tests were first performed before *LANDesk* was loaded to obtain a baseline figure, shown against letter A in the summary table. Then the NLM was loaded, on-access checking enabled, and the tests were run again giving the figures shown at C. Two types of on-demand scan were triggered in turn, and the tests were executed twice more to give the information shown at D and E.

Next, on-access scanning was disabled, and the same tests were performed again as for C, D, and E above, giving F, G, and H. Finally, *LANDesk* was unloaded, and the tests performed a final time, to give the information at B. It should be noted that this information is not the same as that given at A, because unloading *LANDesk* did not unload all the support NLMs which were loaded with it (Btrieve and the Alert Management System).

The time difference between having the NLM loaded (F) and not loaded (A) may be due to the difference in server memory available for the NCOPY program to use as buffers. The apparent anomaly of the overhead being worse when the NLM is unloaded could be due to a hole being left in the server's memory, which is not recovered by system. This may reduce the memory available for buffer space.

Real-time scanning creates an overhead when running, but this has less of an impact than a full scan as only specific files will be checked. This leads to the conclusion that normal operation would be to have real-time scanning, running the full scan only out of hours. This may explain the absence of an hourly scheduled option.

Conclusion

Preparation for installation can be protracted with the amount of files involved. It may also be necessary to go through testing to check for any clashes with other network-based products sharing the same resources, particularly Btrieve – but this would be true for any network product.

It would useful also to have a progress meter on the server display, as well as on the workstation, when scanning. The software provides management tools for remote support of servers and workstations for virus checking, alert management and scan list updating.

The scanner's detection performance has now also improved to a point where it can be considered as a primary virus protection for a network.

LANDesk Virus Protect

Detection Results[1]

In the Wild	270/286	94.4%
Standard	254/265	95.9%
Polymorphic	3805/5500	69.2%

Overhead of On Access Scanning

Time to copy 63 EXE files; 4.6MB (average time in seconds for ten tests).

	Time	% overhead
A) NLM not loaded	11.1	n/a
B) NLM unloaded	20.2	82%
NLM loaded, on access scanning on		
C) No immediate scan	26.6	139%
D) Immediate scan only	45.7	312%
E) Full chksum, CDS, scan	52.5	373%
NLM loaded, on access scanning off		
F) No immediate scan	19.4	75%
G) Immediate scan only	30.3	173%
H) Full chksum, CDS, scan	42.4	282%

Technical Details

Product: *LANDesk Virus Protect v3.0* (update pattern 116).

Developer/Vendor: *Intel Corporation*, 5200 N.E. Elam Young Parkway, Hillsboro, Oregon 97124-6497. Tel +1 503 629 7354, fax +1 503 629 7580, BBS: +1 503 264 7999, World Wide Web: http://www.intel.com/.

Distributor UK: *Intel Corporation*, Piper's Way, Swindon, Wiltshire SN8 2BS. Tel +44 1793 696000, fax +44 1973 444447.

Price: Per server, with quarterly updates: 1 – £699; 4 – £2099; 20 – £6990.

Hardware Used: Server – *Compaq Prolinea 590*; 16MB RAM, 2 GB Disk, *NetWare 3.12*. Workstation – *Compaq 386/20e*, 4MB RAM. 207 MB Disk DOS 6.22, *Windows 3.1*.

[1]Test-sets: For a full listing of the viruses used in this review, see *VB* March 1996, p.20.

PRODUCT REVIEW 2

ThunderBYTE

Dr Keith Jackson

This is a product describing itself as 'the most complete anti-virus system available'. *ThunderBYTE* is a 'multi-package', offering scanner, checksummer, disinfection, memory-resident software, *Windows* components, and other utilities.

It is a well-established product which I have reviewed for *VB* twice before [*September 1993 p.20; September 1991, p.21*]. Many of *ThunderBYTE's* features are network-aware: a network-based review is in the offing.

Documentation

The product's documentation has always been good, and this latest version is no exception. Not only is a 193-page A4 book provided, but a 400 KB documentation file is supplied on diskette. The printed documentation, however, contains only a skimpy index containing little information.

All the *ThunderBYTE* functions are well explained (in both versions of the documentation), and sections are included which describe how to recover from a virus infection, and what strategy to use to avoid viruses in the first place.

Installation

The product was provided for review on a 1.44 MB floppy disk, containing both DOS and *Windows* versions. As the package provided for review was a demonstration version, installation may differ from what is normally available.

Installation is straightforward. For the DOS version, merely specify a subdirectory for the product's files; installation then copies files, updates checksum databases, asks if it should modify AUTOEXEC.BAT to execute the product's memory-resident programs at boot time, scans local drives, and explains printing out the manual. The DOS components of the product occupy 776 KB hard disk space, across 42 files.

The procedure for *Windows* provides various types of installation, but once going, performs much the same tasks as the DOS version. It is necessary to reboot *Windows* to complete the installation: when this is done, a window appears onscreen which continually displays details of the latest scan. The *Windows* components of *ThunderBYTE* occupy 1.51 MB of hard disk space, across 64 files.

Operation

The individual DOS utilities can execute as stand-alone DOS programs, or can be executed from within a shell program driven by drop-down menus. The latter should prove much more productive for naïve users.

The *ThunderBYTE* scanner claims to be both a 'signature' and a 'heuristic' scanner, and will also verify a file's checksum. When applying heuristic methods, the scanner disassembles and analyses files, looking for suspicious instructions – this enables it to detect viruses as yet unknown. Such a detection method is quite acceptable as long as it does not result in false positives (see below).

Scanning Speed

In its default state, the DOS version of *ThunderBYTE* scanned the hard disk of my test computer (393 files, 25.8 MB, 861 files) in 33 seconds. With 'Quick Scan' enabled, this time reduced to 28 seconds. Removing the boot sector scan and memory scanning reduced it further, to 26 seconds. Timings were measured without enabling the product's log file, but enabling it only added about a second to the scan time.

To put these into perspective, *Dr. Solomon's AVTK* scanned the hard disk of my test PC in 5 minutes 1 second, and for the same scan, *Sophos' Sweep* took 7 minutes 50 seconds. Neither of these are slowcoaches, so *ThunderBYTE's* DOS scanner can only be classed as phenomenally quick – it has consistently been the fastest scanner around for several years.

However, there is a problem – the product's *Windows* version required 2 minutes 3 seconds to scan the same hard disk in default mode, and 1 minute 38 seconds under 'Quick Scan'. *Dr. Solomon's AVTK* can scan almost as fast under *Windows* as in its DOS version, so though *ThunderBYTE's Windows* version is still faster than its competitors, the gap has narrowed. The blinding speed of the DOS scanner seems to have been surrendered to *Windows* glitz; however, *ESaSS* states that speed was not its primary concern when developing the *Windows* version of the product.

When *ThunderBYTE's* DOS scanner is executed under *Windows*, it can still scan the hard disk of my test PC in 35 seconds, showing clearly that the product's *Windows* version is slower than its DOS equivalent – it is not merely the presence of *Windows* slowing things down.

The simple and clear DOS menuing system through which all the individual utilities can be controlled.

Windows Version

The *Windows* version of *ThunderBYTE* is unusable without a mouse. Although many features can be accessed via drop-down menus, there seems to be no way without a mouse to activate onscreen buttons, which are the only way to start operation of the scanner and/or the checksummer. Why not?

The E, N, and R keys next to the onscreen buttons are underlined, so following the usual *Windows* conventions, they should be accessible using the Alt key. However, it appears to do nothing here. Even the Tab/Ctrl-Tab key combinations are no use. This is perhaps no problem for desktop PCs, but many laptop owners, Luddites and typists (and I come into all three categories!) still rely on the keyboard. With *ThunderBYTE*, I'm stuffed…

ThunderBYTE for *Windows* twice locked up during a scan. The 'Stop' button was greyed out and only a boot would persuade it to recommence. Things were no better when TBSETUP was executed under *Windows*: it locked up, offered buttons to 'Close' or 'Ignore' the error produced, and then gave a GPF error. *Windows* then required a reboot before operations could proceed.

I tried TBSETUP again but it exhibited the same problem. On both occasions when TBSETUP crashed, it 'lost' all the previously selected setup information, and all selections seemed to return to their default states. The DOS version worked fine, and initialised the checksums correctly.

Scanner Detection

I tested the virus detection capability of the product's DOS version against the test-sets described in the Technical Details section below. To prevent accidental infection, I usually store the test-sets as renamed (to *.CO and *.EX) files – this simply makes it impossible to run them accidentally. However, when I ran *ThunderBYTE* against the viruses in this state (having reconfigured it to check these extensions), the detection rates were suspiciously low.

So, the virus files were renamed to the standard executable extensions; *.COM, *.EXE, *.DOC and *.DOT (the latter two containing samples of the *Word* virus Concept). In this (much more realistic) circumstance, the detection rate of the scanner improved dramatically, and it is these figures that are quoted here.

ThunderBYTE was note-perfect against the In the Wild test-set; an impressive 286 out of 286 detected. In the Standard test-set, only three viruses were missed: the two samples of Cruncher and one of Argyle, giving a detection rate of 99% (301 out of 304).

In the Polymorphic test-set, 6553 samples were pronounced infected in a test-set of 7500; giving a detection rate of 87%. Of the different viruses contained within (there are 500 samples of each of the fifteen viruses), *ThunderBYTE* fared worst against MTZ.4510, flagging only one file as 'probably infected'. Of the other sets, 329 PeaceKeeper.Bs, 407

In the software provided for review, the Virus Information file is present only in the *Windows* version of the product.

DSCE.Demos, 440 Code.3952:VICE.05s, 454 SMEG_v0.3, and 461 of both Girafe:TPE and Coffeeshop were the only incomplete scores – a good result all round.

All of these figures were produced using the default settings of the product: *ThunderBYTE* allows the user to increase or decrease the sensitivity of its heuristics.

The reason behind the fact that the product will detect viruses differently in .COM files than in .CO files would appear to be something to do with the heuristic features which are part of the product.

There is no way to identify that a file is definitely a COM file other than by looking at its extension (as opposed to EXE files, which have an easily-identifiable header). If the product were to attempt to emulate data files, it would run into terrible false positive problems, as it is very easy for data to look like a decryption loop.

As far as boot sectors go, of the twenty used, *ThunderBYTE* missed only Peanut. Actually, even this should count as partially detected, as the *Windows* version of the product said that the diskette was infected, but omitted to make an entry in the log file. *ThunderBYTE's* DOS version appeared to think that the disk was clean – a curious inconsistency.

Detection Problems

ThunderBYTE for *Windows* has problems handling its log files: the undetected boot sector virus mentioned above is not the only one. On large log files, if settings other than the default are selected, the *Windows* version loses the summary data normally written at the end of a log file after a scan has finished. This happened when 'High Heuristic' scanning or 'Quick Scan' was enabled. There are probably other choices that exhibit this problem, but I did not test this any further.

Last time *ThunderBYTE* was reviewed, it produced no false positives: this time around, the same result was received, but only in the default mode of operation.

With 'High' heuristics enabled, of 306 files (from a total of 861) checked on the hard disk of my test computer, four were found to contain a 'code decryption routine or

debugger trap', three had 'Suspicious Memory Allocation' code, five had an 'Inconsistent exe-header', seven had a 'Suspicious file access', two had 'Garbage instructions', and two were described as having 'Stealth capabilities'.

In all, there were 48 other suspicious instances reported – and this on a hard disk containing no viruses whatever. *ThunderBYTE* even thought that one of its own files (TBGENSIG) could be suspicious!

Checksummer Database

The *ThunderBYTE* documentation file goes on at length to try and explain why *ThunderBYTE* creates an Anti-Vir.Dat file in every single subdirectory, rather than creating a single reference file in its own subdirectory. When summarised, the stated reasons are that it is more intuitive; that if the subdirectory is moved/deleted, the data file follows it; and finally, that such a scheme is easier to maintain (especially on networks).

No matter how this is all disguised, it still looks like excuses. In fact, the developers could have maintained a single file, but chose not to. Put bluntly, it is *my* hard disk: any product that scatters files in every subdirectory will continue to get short shrift from me.

Memory-resident Software

ThunderBYTE has an intriguing, structured approach to memory-resident software. A 'driver' program must first be loaded; then every extra facility which is added uses the features provided by the driver program, and only adds slightly to the amount of memory required.

The memory-resident features which are available with *ThunderBYTE* are a background scanner which automatically tests files being executed and/or copied, a checksummer which checks every file before it is executed, a memory checker which detects any attempt by a program to remain memory-resident, a file checker which prevents executable programs from infecting other programs, and a disk guard program which detects any direct (i.e. not via DOS) write to disk and any attempt to format a disk.

The chosen memory-resident features are included in a special batch file (TBSTART.BAT) which is included in the AUTOEXEC.BAT file installed by *ThunderBYTE*.

I measured the overhead imposed by the memory-resident components of the product by copying 40 files (1.25 MB) from one subdirectory to another. With none of its memory-resident software present, the task took 22 seconds to perform – this rose to 58 seconds when the background scanner was present. The other memory-resident components did not add much to this measured overhead.

When only the driver software is memory-resident, 3.4 KB of memory is occupied. The various memory-resident features provided with *ThunderBYTE* each add between 800 bytes and 1.4 KB to overall memory requirements.

The Rest

Utilities are provided to restore the original (uninfected) boot sector, CMOS and Partition Tables. The partition table can (if required) be replaced with a *ThunderBYTE* version which claims to offer greater resistance to viruses.

ThunderBYTE also claims to be able to 'clean' viruses from infected files, but this has not been reviewed. Infected files should be replaced with copies that are known to be clean: to that end *ThunderBYTE* provides a utility which positively erases a file by overwriting every byte with zeros.

The DOS version of the product does not provide any on-line 'Virus Information'. This database is only available to Registered Users (remember, I am reviewing a 'Demonstration Version'). Curiously, the *Windows* version, which is installed from the same disk, includes this 'Virus Information'.

Conclusions

ThunderBYTE is very quick indeed at scanning for viruses, probably still the fastest around, but unfortunately this only applies to the DOS version. The *Windows* version is slower, and contains many bugs. This is a shame, as I still admire the DOS version, and recommend it on the grounds of fantastic speed and very good detection rates.

It is a sobering thought that, in my last review of the product two years ago, I wrote: 'I commend the developers of *ThunderBYTE* for ignoring the trend toward making anti-virus software into beautifully-sculpted *Windows* programs'. Now look what's happened. My new name is Nostradamus.

ThunderBYTE has always been a set of utilities; an excellent set of utilities. Such a structure does not sit kindly in the *Windows* view of the world, and it looks as if its developers are having a tough time making necessary changes. However, given *ThunderBYTE's* excellent track record, it is likely that the bugs in the *Windows* software will soon be fixed.

Technical Details

Product: *ThunderBYTE v7.0* (no serial number visible).

Developer/Vendor: *ESaSS BV*, Saltshof 10-18, NL-6604 EA Wijchen, The Netherlands. Tel +31 24 642 2282, fax +31 24 645 0899, email info@thunderbyte.com.

Availability: DOS components require 0.8 MB hard disk space, 256K RAM, and DOS v3.0 or above. *Windows* components require an 80386 processor, DOS v5.0 or above, *Windows v3.1*, 1 MB RAM and 1.5 MB hard disk space.

Price: Applies to *TBAV* for DOS, *Win3.xx*, *Win 95*, *NT*, and *OS/2*. Single-user licence: HFl 244; 1-5 users: HFl 496; 6-10 users: HFl 880; 11-25 users: HFl 1560; 26-50 users HFl2680; 51-100 users: HFl 4600; 101-200 users: HFl 7990. Includes bi-monthly updates. Larger licence prices on request.

Hardware used: *Toshiba 3100SX*; a 16 MHz 386 laptop with one 3.5-inch (1.4 MB) floppy disk drive, a 40 MB hard disk and 5 MB RAM, running under MS-DOS v5.00 and *Windows v3.1*.

Virus test-sets: For a complete listing of the Boot Sector test-set, see *VB* March 1996 p.23. The Standard, In the Wild, and Polymorphic test-sets are listed in detail in *VB* April 1996 p.20.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England

Tel 01235 555139, International Tel +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email editorial@virusbtn.com
CompuServe address: 100070,1340
World Wide Web: http://www.virusbtn.com/

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel +1 203 431 8720, fax +1 203 431 8165

END NOTES AND NEWS

***VB 96*, the international virus prevention conference,** will be held in Brighton, UK, on 19–20 September 1996. Details from conference assistant Alie Hothersall; Tel +44 1235 555139, fax +44 1235 531889.

Data Fellows*, European distributor of *F-Prot, has announced the addition of Peter Szor (one of the developers of *Pasteur AntiVirus*) as primary virus analyst on its development team. For information on *F-Prot*, Peter Szor, or *Data Fellows*, Tel +358 0 478 444, fax +358 0 478 44 599, or email sales@DataFellows.com.

S&S International has announced the addition of a macro virus detection engine to *Dr Solomon's AVTK*. The engine operates in conjunction with FindVirus and WinGuard). On the learning front, the company is presenting **Live Virus Workshops** at the *Hilton National* in Milton Keynes, Bucks, UK on 13/14 May 1996. Details from the company: Tel +44 1296 318700, fax +44 1296 318777.

First.Base is hosting a series of IT security and Internet workshops in Sussex, UK, throughout the next two months. Sessions will include **Internet security (incorporating defence against viruses)** and disaster contingency planning. Information can be obtained from *First.Base* on Tel +44 1903 879879, fax +44 1903 879274.

The next anti-virus workshops presented by *Sophos Plc* will be on 22/23 May 1996 at the training suite in Abingdon, UK. The seminar costs £595 + VAT; one single day, £325 + VAT (day one: Introduction to Computer Viruses; day two: Advanced Computer Viruses). **The latest addition, supported by *Microsoft UK*, covers macro virus detection and removal**. Contact Julia Line on Tel +44 1235 544028, fax +44 1235 559935, or visit http://www.sophos.com/ for information.

Precise Publishing Ltd will be holding more **Live Virus Workshops** (15 May 1996, 12 June 1996, 17 July 1996). Details are available from the company; Tel +44 1384 560527, fax +44 1384 413689.

Reflex Magnetics has several courses coming up: **Live Virus Experiences** (12/13 June, 9/10 October), The Hacking Threat (24-26 July), Internet Security and Firewalls (30 May, 22 July), and DTI Security Codes of Practice (31 May). For further information, contact Rae Sutton: Tel +44 171 372 6666, fax +44 171 372 2507.

From 3–5 June 1996, the *Computer Security Institute* (*CSI*) will be sponsoring ***NetSec 96***. The conference, to be held in San Francisco, will focus on security issues, problems, and solutions in networked environments. Further details, and a free catalogue, are available from the *CSI* via email at csi@mfi.com, or Tel +1 415 905 2626, fax +1 415 905 2218.

McAfee Associates announced on 10 April 1996 that **it has acquired yet another company**. Continuing its diversification, the company has bought out *Vycor Corporation*, which develops and sells client/server help-desk solutions. This is *McAfee's* fourth acquisition in 24 months: information on all the buy-outs is available from *McAfee* on Tel +1 408 988 3832, fax +1 408 970 9727.

The Fourth International Conference on Information Warfare will take place in Brussels, Belgium on 23/24 May 1996. For information, contact the *NCSA* (*National Computer Security Association*), 10 South Courthouse Ave, Carlisle PA 17013, USA. Tel +1 717 258 1816, fax +1 717 243 8642, email conference@ncsa.com.

It has been brought to *VB's* attention that various companies and organisations have, without our knowledge or permission, been reproducing information published in *Virus Bulletin*. Although we are always pleased to help with reproduction requests, all material in these pages is the copyright of *VB*: reproduction must be authorised in writing, and the final production proof approved by *VB*, before release. Information on reproduction is available from our offices.